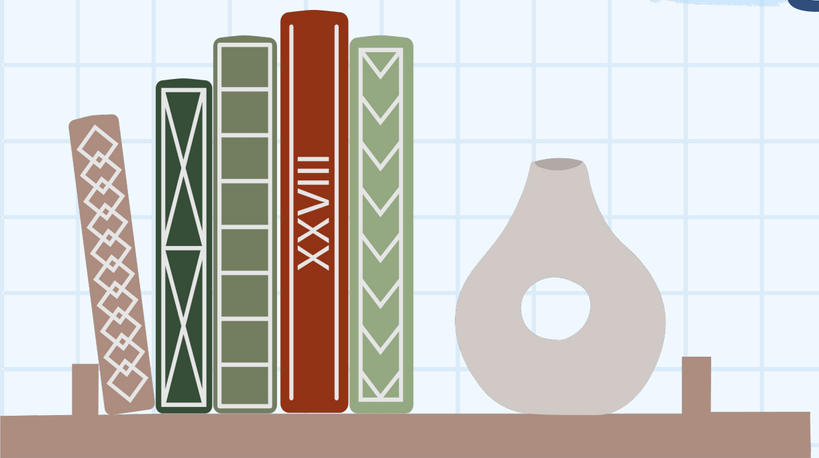
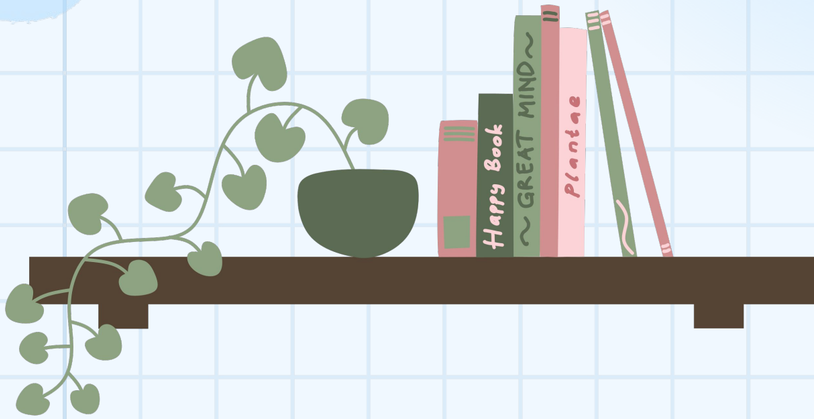
BS./BSC.IN

Applied AI and Data Science

Algorithmic Thinking & its Applications





Powered by Futurense

Problem-solving Process

- Problem analysis
- Alternative consideration
- Choosing an approach
- Problem decomposition
- Algorithm development
- Algorithm correctness
- Algorithm efficiency
- Reflection

Problem Understanding

Is the step that leads to the identification of the problem characteristics.

For algorithmic problems, it starts by recognizing the input categories of the problem and the selection of the required output category for each input category.

Basic strategies

Identify outputs

02

Identify inputs

01

If yes, how;
If not, why?

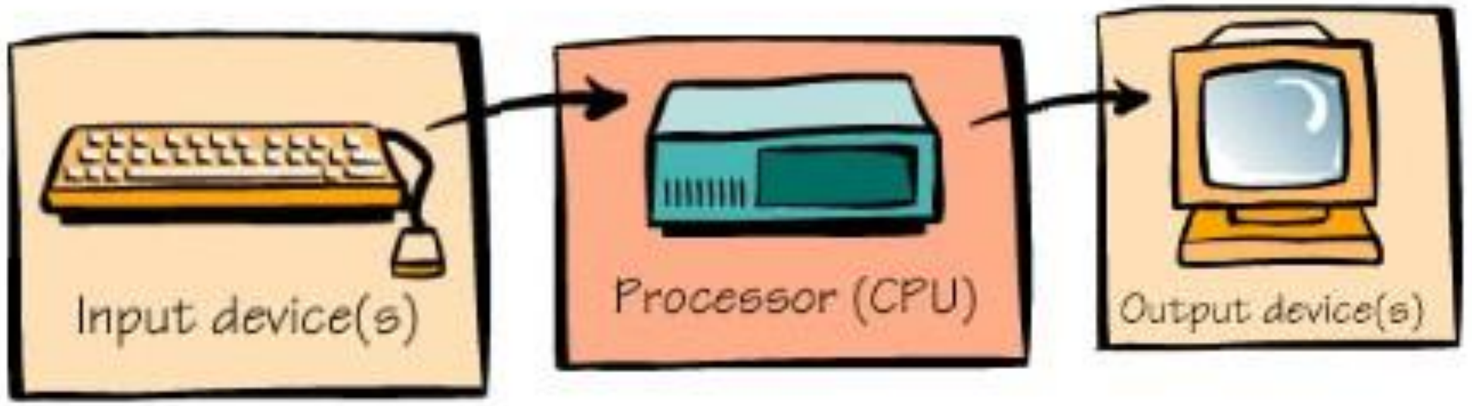
03

Each input is a subtask

04

Extreme cases

What Computers Do



Receive input

Process Information

Produce Output

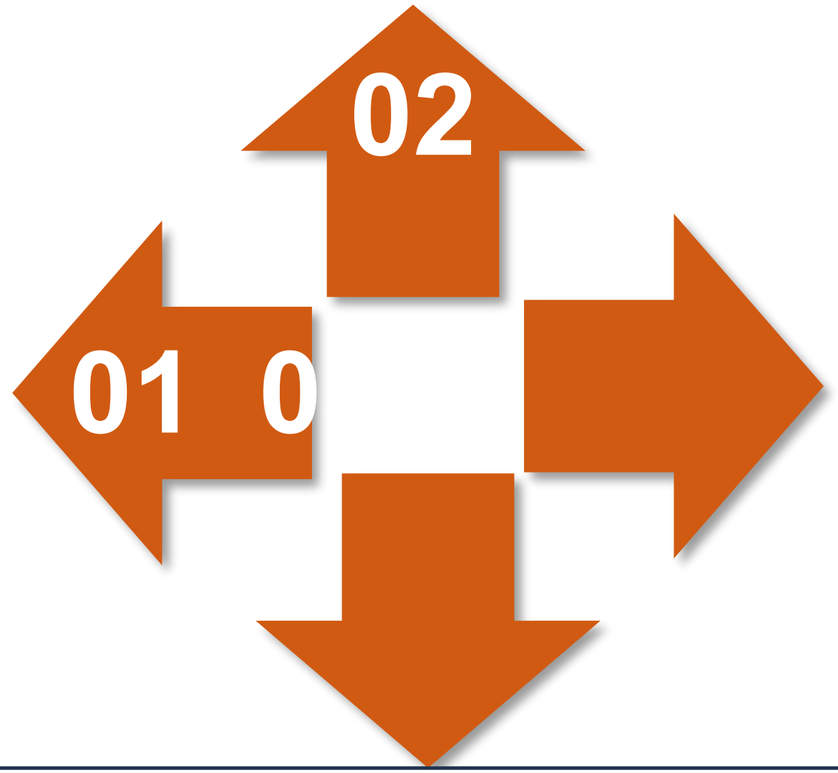
Solution Design

Is the major challenge facing novice learners.

The examination of a given problem’s inputs and outputs clarifies the problem to the problem solver. The next stage is to define the variables needed to solve the problem.

Basic Strategies

Stepwise refinement



Define problem variables

Identify algorithmic patterns

Variables

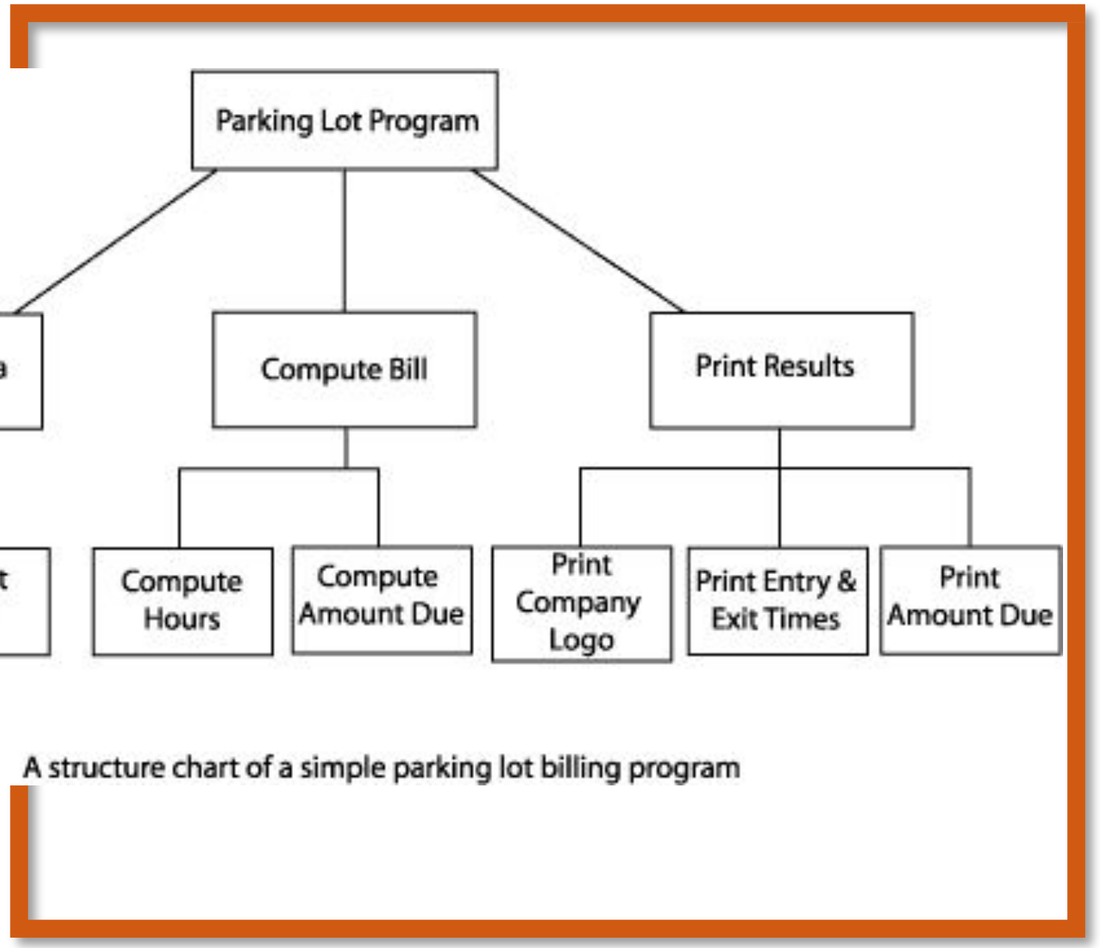
A variable can be thought of as a box that can hold one value at a time.

Stepwise Refinement

- Is a top-down methodology that first obtains the overview of the structure of the problem and the relationship among each parts, and then to address specific and complex issues related to the implementation of subpart.

- refers to the progressive refinement in small steps of a program specification into a program.

Step-wise (top-down) Refinement



A structure chart of a simple parking lot billing program

- was used first in the paper titled *Program Development by Stepwise Refinement* by Niklaus Wirth, the author of the programming language Pascal and other major contributions to software design and software engineering, in the *Communications of the ACM,* Vol. 14 (4), 1971, pp. 221-227.
- Wirth said: "It is here considered as a sequence of design decisions concerning the decomposition of tasks into subtasks and of data into data structures."

Quiz

- The minimum number of steps m to find the sum and product of two given numbers.
  - 1
  - 3
  - 4
  - 5

Quiz

- The minimum number of steps m to find the sum and product of two given numbers.
  - 1
  - 3
  - 4
  - 5

Stepwise

- Start with the initial problem statement
- Break it into a few general steps
- Take each "step", and break it further into more detailed steps
- Keep repeating the process on each "step", until you get a breakdown that is pretty specific, and can be written more or less in pseudocode
- Translate the pseudocode into real code

Stepwise Refinement (Example)

Problem Statement:
Determine the class average for a set of test marks, input by the user. The number of test marks is not known in advance (so the user will have to enter a special code -- a "sentinel" value -- to indicate that he/she is finished typing in grades).

Initial breakdown into steps

- Declare and initialize variables
- Input marks (prompt user and allow input)
- Compute class average and output result

Now, breaking down the "compute" step further, we got:

Compute:

- ✔ add the marks
- ✔ count the marks
- ✔ divide the sum by the count

Revised breakdown of steps

- Declare and initialize variables
- Input marks -- count and add them as they are input
- Compute class average

Breaking the steps into smaller steps

So, now we can break down these 3 steps into more detail. The input step can roughly break down this way:

loop until the user enters the sentinel value (-1 would be good)

- prompt user to enter a mark (give them needed info, like -1 to quit)
- allow user to type in a mark (store in a variable)
- add the mark into a variable used for storing the sum
- add 1 to a counter (to track how many grades)

Breaking the steps into smaller steps

We could specifically write this as loop. So one more refining step would be a good idea, to formulate the pseudo-code more like the actual code we would need to write. For example:

do
- prompt user to enter a mark (give them needed info, like -1 to quit)
- allow user to type in a mark (store in a variable)
- add the mark into a variable used for storing the sum
- add 1 to a counter (to track how many grades)

while user has NOT entered the sentinel value (-1 would be good)

Breaking the steps into smaller steps

If we look at this format, we realize that the "adding" and "counting" steps should only be done if the user entry is a grade, and NOT when it's the sentinel value. So we can add one more refinement:

do
- prompt user to enter a mark (give them needed info, like -1 to quit)
- allow user to type in a makr (store in a variable)
- if the entered value is a MARK (not the sentinel value)
  add the grade into a variable used for storing the sum
  add 1 to a counter (to track how many grades)

while user has NOT entered the sentinel value (-1 would be good)

Breaking the steps into smaller steps

This breakdown helps us see what variables are needed, so the declare and initialize variables step can be now made more specific:

initialize variables:

- a mark variable (to store user entry)
- a sum variable (initialized to 0)
- a counter (initialized to 0)

Compute and print:

- divide sum by counter and store result
- print result

Putting it all together

initialize variables:
  a mark variable (to store user entry)
  a sum variable (initialized to 0)
  a counter (initialized to 0)

grade entry:
  do
    prompt user to enter a mark (give them needed info, like -1 to quit)
    allow user to type in a mark (store in a variable)
    if the entered value is a MARK (not the sentinel value)
      add the mark into a variable used for storing the sum
      add 1 to a counter (to track how many grades)
  while user has NOT entered the sentinel value (-1 would be good)

Compute average:
  divide the sum by the counter
  print the answer

Thank you